1.1 CSS Syntax

CSS (Cascading Style Sheets) is used to style HTML elements. A basic CSS rule consists of:

```css
selector {
  property: value;
}
```

- **Selector**: Specifies which HTML element(s) to style. Examples include `p` for paragraphs or `h1` for headings.
- **Property**: Defines the aspect of the element to style (e.g., `color`, `font-size`, `margin`).
- **Value**: Sets the specific value for the property.

Example:

```css
h1 {
  color: blue;
  font-size: 24px;
}
```

This rule changes the color of all `<h1>` elements to blue and sets their font size to 24 pixels.

1.2 CSS Selectors

Selectors define which elements the styles apply to. Common types include:

- **Universal Selector (`*`)**: Targets all elements on the page. Useful for applying global resets.

  ```css
  * {
    margin: 0;
    padding: 0;
  }
  ```

- **Type Selector**: Targets specific HTML tags. For example, applying a font style to all `<p>` elements:

  ```css
  p {
    font-family: Arial, sans-serif;
  }
  ```

- **Class Selector (`.`)**: Targets elements with a specific class. Classes are reusable across multiple elements.

```css
.highlight {
  background-color: yellow;
}
```

- **ID Selector (`#`)**: Targets an element with a specific ID. IDs should be unique per page.

```css
#main {
  text-align: center;
}
```

- **Grouping Selectors**: Apply the same styles to multiple selectors. This helps reduce redundancy.

```css
h1,
h2,
h3 {
  color: navy;
}
```

1.3 Text Styling

CSS allows for precise control over text appearance:

- **Color**: Sets the color of the text. You can use named colors, hex codes, RGB, or HSL values.

```css
p {
  color: gray;
}
```

- **Font Size**: Controls the size of the text. Values can be in `px`, `em`, `%`, or `rem` units.

```css
h1 {
  font-size: 36px;
}
```

- **Font Family**: Specifies the font type, with fallbacks in case the preferred font is unavailable.

```css
body {
  font-family: "Times New Roman", serif;
}
```

- **Text Alignment**: Aligns text horizontally (`left`, `center`, `right`, or `justify`).

```
h1 {
  text-align: center;
}
```

---

2. Box Model

2.1 Box Model Overview

Every HTML element is treated as a rectangular box consisting of four areas:

1. **Content**: The innermost area containing text, images, or other content.
2. **Padding**: Space between the content and the border, adding breathing room inside the element.
3. **Border**: The edge surrounding the padding and content, which can be styled with width, color, and type.
4. **Margin**: Space outside the border, separating the element from neighboring elements.

Diagram:

```
| Margin |
| Border |
| Padding |
| Content |
```

Example:

```
div {
  padding: 10px;
  border: 2px solid black;
  margin: 15px;
}
```

This rule adds 10px padding, a 2px black border, and 15px margin around a `<div>` element.

2.2 Adjusting Box Model Properties

- **Padding**: Adds space inside the element, between content and border.

  ```
  p {
    padding: 20px;
  }
  ```

- **Margin**: Creates space outside the element to separate it from other elements.

```css
div {
  margin: 10px auto;
}
```

`auto` can be used to center block elements horizontally.

- **Border**: Defines the style, width, and color of the border.

```css
img {
  border: 5px solid red;
}
```

- **Width and Height**: Sets the dimensions of an element. Default values are determined by the content.

```css
.box {
  width: 200px;
  height: 100px;
}
```

---

3. Responsive Design

3.1 Media Queries

Media queries enable you to apply styles based on specific conditions, such as screen width or orientation. This is essential for creating mobile-friendly designs.

Example:

```css
/* For screens wider than 768px */
@media (min-width: 768px) {
  body {
    font-size: 18px;
  }
}

/* For screens narrower than 768px */
@media (max-width: 768px) {
  body {
    font-size: 16px;
  }
}
```

3.2 Flexible Layouts

Combine percentage-based widths with media queries to create layouts that adapt to different screen sizes.

Example:

```css
div {
  width: 50%;
  margin: 0 auto;
}

@media (max-width: 600px) {
  div {
    width: 90%;
  }
}
```

This example sets a 50% width for larger screens and adjusts to 90% for smaller screens.

Learning Activities

Activity 1: Hands-on Styling Tasks

1. Style a basic HTML document by:
   - Changing text color and alignment.
   - Adding padding, margins, and borders to elements.
   - Using media queries to adjust text size for smaller screens.

Activity 2: Small Projects

1. Create a simple webpage featuring:
   - A title and heading styled with CSS.
   - A paragraph with custom font size and color.
   - A box containing an image, styled with padding, borders, and margins.

Activity 3: Live Coding Demonstrations

- Demonstrate the box model using borders, padding, and margins.
- Show how to apply media queries for responsive design.

Deliverable: Mini-Project

Project Title: Styled Personal Webpage

Description: Enhance the personal webpage created during Weeks 2–3 with CSS. The styled version should include:

- **CSS Fundamentals:**
  - Use selectors to style headings, paragraphs, and lists.

    - Apply custom text colors, fonts, and alignment.

- **Box Model:**

    - Add padding, borders, and margins to create visually appealing layouts.
    - Define specific dimensions for images and content sections.

- **Responsive Design:**

    - Incorporate media queries to ensure the page looks good on different screen sizes.

Submission:

- Submit the HTML and CSS files.
- Ensure proper formatting, indentation, and use of CSS best practices.
